OIPE JC184 DEC 23 2004 PATENT & TRADEMARK OFFICE

AF/2123 IFW $

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: Timothy M. ASKINS et al.	Attorney Docket No.: CNTRP001
Application No.: 09/496,563	Examiner: GRAIG, Dwin M.
Filed: February 2, 2000	Group: 2123
Title: MULTI-THREADED FRAME SAFE SYNCHRONIZATION OF A SIMULATION	

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the U.S. Postal Service with sufficient postage as first-class mail on December 20, 2004 in an envelope addressed to the Commissioner for Patents, Mail Stop Appeal Brief-Patents, P.O. Box 1450 Alexandria, VA 22313-1450.

Signed: [illegible]
Laura M. Dean

APPEAL BRIEF TRANSMITTAL
(37 CFR 192)

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

12/27/2004 RFEKADU1 00000024 09496563
02 FC:2402 250.00 OP

This brief is in furtherance of the Notice of Appeal filed in this case on .
This application is on behalf of

☒ Small Entity ☐ Large Entity

Pursuant to 37 CFR 1.17(f), the fee for filing the Appeal Brief is:
☒ $250.00 (Small Entity) ☐ $500.00 (Large Entity)

☒ Applicant(s) hereby petition for a three month extension(s) of time to under 37 CFR 1.136.

If an additional extension of time is required, please consider this a petition therefor.

☐ An extension for months has already been secured and the fee paid therefor of $ is deducted from the total fee due for the total months of extension now requested.

☐ Applicant(s) believe that no (additional) Extension of Time is required; however, if it is determined that such an extension is required, Applicant(s) hereby petition that such an extension be granted and authorize the Commissioner to charge the required fees for an Extension of Time under 37 CFR 1.136 to Deposit Account No. 500388.

12/27/2004 RFEKADU1 00000024 09496563
01 FC:2253 510.00 OP

Total Fee Due:

Appeal Brief fee	$250.00
Extension Fee (if any)	$510.00
Total Fee Due	$760.00

☒ Enclosed is Check No. 24877 in the amount of $760.00.

☒ Charge any additional fees or credit any overpayment to Deposit Account No. 500388, (Order No. CNTRP001). Two copies of this transmittal are enclosed.

Respectfully submitted,
BEYER WEAVER & THOMAS, LLP

[illegible]

Alan S. Hodes
Reg. No. 38,185

P.O. Box 70250
Oakland, CA 94612-0250
(650) 961-8300

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF APPEALS

EX PARTE TIMOTHY M. ASKINS ET AL.

Application for Patent

Filed February 2, 2000

Application No. 09/496,563

FOR:

MULTI-THREADED FRAME SAFE SYNCHRONIZATION OF A SIMULATION

APPEAL BRIEF UNDER 37 CFR 41.37

CERTIFICATE OF MAILING
I hereby certify that this correspondence is being deposited with the U.S. Postal Service with sufficient postage as first-class mail on December 20, 2004 in an envelope addressed to the Commissioner for Patents, P.O. Box 1450 Alexandria, VA 22313-1450.

Signed: [illegible]
Laura M. Dean

BEYER WEAVER & THOMAS, LLP
Attorneys for Applicant

TABLE OF CONTENTS

I. REAL PARTY IN INTEREST

The real party in interest is Centric Software, Inc., the assignee of the present application.

II. RELATED APPEALS AND INTERFERENCES

The undersigned is not aware of any related appeals and/or interferences.

III. STATUS OF CLAIMS

Claims 1-14 are pending in this application. Claim 1 is an independent claims, and claims 2-7 depend either directly or indirectly on independent claim 1. Claim 8 is an independent claim, and claims 9-14 depend either directly or indirectly on independent claim 8.

Claims 1-14 stand rejected.

Claims 1-14 are appealed.

IV. STATUS OF AMENDMENTS

No amendments have been filed subsequent to final rejection.

V. SUMMARY OF CLAIMED SUBJECT MATTER OF THE INDEPENDENT CLAIMS

Independent claim 1

The summary provided herewith refers to the specification, including the figures, but the provision of this summary is not intended to affect, by itself, the scope of the claims, including to limit the claim scope to the embodiments disclosed in the specification.

The subject matter recited in independent claim 1 relates generally to the field of a computer program product to configure a computer to execute a simulation of an environment. Various aspects of the simulation are executed asynchronously, but each asynchronously-executed aspect has a coherent view of the simulated environment.

The system of claim 1 includes three primary elements – a plurality of **service program code means** operating in a **rate independent** manner (shorthand for **executing at a rate independent of the other service program code means**) to determine simulated attributes of mechanical systems operating in a simulated environment; **write queue code means** associated with each service program code means to queue write requests to write simulated, determined attributes to an object database, and **node means** that coordinates execution of the queued write requests to the object database so that each service program code means has a coherent view of the object attributes.

The interoperability of the elements, including with respect to time, is a significant feature of the invention recited in claim 1. The description in the patent application first discloses the elements, in Figures 1-4, somewhat in operational isolation (at least, in temporal isolation) and then goes on to describe the interoperability of the elements in the timelines of Figures 5 and 6.

Referring first to Figure 1, element 102 in Figure 1 is an example of the **object database**. As discussed at page 4, lines 2-5: "The 'attributes' of the 'objects' in the database 102 represent the 'state' of the environment being simulated."

The **service program code means** correspond, for example, to the service execution block 104 in Figure 1. See page 4, lines 5-7: "Block 104 illustrates execution of the 'service' (in this case, service A through C) that constitute the active (computational) part of the simulation." As discussed at page 4, lines 17-22, the behavior context 108 in Figure 1 is "the programming employed by a service (in the service execution block 104) that defines how a service is to affect the modeled environment during a simulation."

While Figure 1 provides an overview of the computer program product, Figure 1 does not explicitly show **each service program code means executing at a rate independent of the other service program code means**. Figure 1 also does not explicitly show the **write queue program code means** or the **node program code means**.

Figure 2 illustrates an example of phases of processing by a typical service, in isolation (i.e., without illustrating how the service interoperates with other services), to perform what may be considered in some sense to be a subset of the simulation. The Phase 1 Computation processing 202 is defined by the behavior context 108 (Figure 1). As disclosed at page 6, lines 10-13, during the Phase 1 Computation processing 202, the **write queue program code means associated with each service program code means** "queues up (but does not yet process) 'change requests.' A 'change request' is a request to a node to change the attribute of an object represented by the node." The Phase 2 Process Change Requests processing 204 illustrates the operation of the service to cause the **node program code means** to process the queued write requests. See page 6, lines 17-23.

Figures 3 and 4 illustrate an example of the operation of the **node program code means** to coordinate execution of the queue write requests. See page 7, line 1 to page 8, line 10. As disclosed at page 5, lines 10-14,

> The node interacts with services to provide frame safe access to its data members. Each node may maintain multiple 'node images' for an object (or for a portion of an object). Each service potentially sees different sets of changes at various times, although all services are ultimately working on the same values.

It is these "node images" referred to at page 5, lines 10-14 that are being manipulated in Figures 3 and 4. By this manipulation, **each service program code means has a coherent view of all the object attributes**. This "coherent view" aspect is perhaps more apparent from Figure 5 and 6, as is the rate independent aspect.

Figures 5 and 6 are each a timeline illustrating an example how the elements of the program recited in independent claim 1 operate together. In Figure 5, the rate independence of the services may not be completely apparent, while Figure 6 emphasizes the rate independence of the services. As discussed at page 8, lines 14-21,

> Fig. 5 is a timeline that shows how each service executes in its own "frame time". In Fig. 5, the actual time a particular service spends in a frame differs from the time spent in a

frame by another service, although this isn't shown in Fig. 5. That is, the services are not synchronized as would appear from Fig. 5 on its face. In fact, though, as is shown later with reference to Fig. 6, it appears to each service that it is in fact operating synchronously with the other services.

In other words, Figure 5 emphasizes how each service has a coherent view of all the object attributes, *in spite of* the rate independence of the services. For example, it could perhaps be considered that, to each service, it appears to be operating synchronously to the other services. Figure 6, on the other hand, illustrates a "higher up" view from which it can be seen that, even though the services are operating in a rate independent manner, the services each have a coherent view of the object database

As mentioned above, Figure 6 more clearly shows the rate independent nature of the service program code means. That is, each service (Service A, Service B and Service C) operates according its own rate. The cached write operations (by the write queue computer readable code means) result in cached node images and current node images under the control of the node computer readable code means.

Independent claim 8

Independent claim 8 recites a method that substantially parallels the recitation of the computer program product in claim 1. Thus, a separate explanation specifically directed to the subject matter recited in claim 8 is not provided.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 1-14 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Bershteyn (U.S. Patent No. 5,678,028) in view of Schoening (U.S. Patent No. 6,205,465).

VII. ARGUMENT

The Rejection of Claims 1-14 Under 35 U.S.C. §103(a).

A. *The Examiner has not met the burden of showing that Bershteyn discloses what the Examiner contends Bershteyn discloses, nor does Bershteyn in fact disclose what the Examiner contends Bershteyn discloses.*

The Examiner relies on Bershteyn as a primary reference. The Examiner contends, among other things, that Bershteyn discloses a plurality of service code means, each service code means associated with at least a subset of object attributes in an object data base. The Examiner references Figures 1 and 2; and col. 4, lines 42-67 and col. 5, lines 1-6 as support for this portion of the contentions relative to Bershteyn.

Unfortunately, as Applicant stated in the response filed after final rejection, the Examiner does not specifically point out where and how the cited portion of Bershteyn is contended to disclose the "rate independent feature" recited in the claims. In fact, the Examiner does not even point out how the cited portions of Bershteyn are considered to disclose the "service code means" themselves.

We first discuss the Examiner's contentions with regard to the "service code means" themselves, hampered by the Examiner's lack of explanation of what is considered to be the "service code means." Perhaps the Examiner considers the various "simulators" 12, 14, 16 and 18 (disclosed in Figure 1) to be the "service code means." However, contrary to the Examiner's assertions, these "simulators" do not operate at a rate independent of the rate of the other "simulators." Rather, the execution of each "simulator" is controlled by the simulator manager 20, which "periodically calls different simulators and passes control to these simulators." Col. 5, lines 30-31. "The simulators simulate for a given period of time and then pass control back to the simulation manager, which might call other simulators or prompt the user for more commands." Col. 5, lines 31-35. Thus, each "simulator" does not execute at any "rate" at all, let alone a rate that is independent of the rate at which each other simulator is executing.

In the response to the final rejection, Applicant requested the Examiner to more specifically point out where and how the disclosure of Bershteyn is contended to disclose this feature, but the Examiner only responded that "Applicant's arguments . . . have been found to be unpersuasive." (Applicant recognizes that, perhaps, this may be all that is required of the Examiner procedurally in an advisory action. That said, Applicant still does not have the information necessary to adequately respond to the Examiner's contentions without resorting to speculation as to the Examiner's contentions. For example, Applicant surmised that perhaps the Examiner considers the timers in Bershteyn's Figure 3 to be the "service code means," but the Examiner has not cited Figure 3 in support of a contention that Bershteyn discloses the rate-independent "service code means.")

Furthermore, there does not even appear to be a disclosure in the cited portions of Bershteyn of an object database to which simulated attributes are written. Figure 1 does not show an object database, nor does Figure 2. The cited portions of col. 4 and col. 5 describe Figures 1 and 2, and do not appear to describe a database at all. Perhaps the Examiner considers a disclosure of an object database to be inherent in the cited portions of Bershteyn? If so, the Examiner has neither made, nor properly supported, such a contention. See MPEP 2112, which states "In relying upon the theory of inherency, the examiner must provide a basis in fact and/or technical reasoning to reasonably support the determination that the allegedly inherent characteristic necessarily flows from the teachings of the applied prior art."

B. *The Examiner has not met the burden of showing that Schoening discloses what the Examiner contends Schoening discloses, nor does Schoening in fact disclose what the Examiner contends Schoening discloses.*

The Examiner contends that Schoening discloses coordinating execution of write requests in a situation where execution of portions of a simulation are executing in parallel. However, Schoening (in a portion not cited by the Examiner) clearly discloses that execution in a <u>single thread</u> is required when SMFunctions require coherent access to a data store. See col. 42, lines 57-67.

> Such SMFunctions execute in a single thread because they need access to all members over which the evaluation is proceeding, for example, when computing the topology of a network. Multiple threads executing in parallel are permitted to run when one or more SMFunctions indicate that they can run in parallel (allIsRequired==FALSE). For example, multiple threads can run in parallel when a discovery timeBase is gathering data from devices 102 of a network 100. It is appropriate to run in parallel in this case because the SMFunctions must wait for responses from devices 102 to the requests that the SMFunctions make of the devices.

Therefore, when Schoening would otherwise require coordinated execution of write requests, Schoening "punts" and falls back to single thread processing.

C. The Examiner's cited "motivation to combine" is improper, as well as being unsupported.

It is also respectfully submitted that, in any event, the Bershteyn reference is improperly combined with the Schoening reference. In particular, Applicant respectfully submit that the alleged motivation to combine Bershteyn and Shoening is wholly unsupported. Specifically, the Examiner contends that "An artisan of ordinary skill in the art would have been motivated to seek a method of executing a simulation in a system where the execution can take place in parallel and do so in a manner that would allow for access to simulated data objects in such a manner as to not corrupt the attributes of those objects by more than one executing simulation process trying to write to those object attributes at the same time."

In the first place, while Applicant requested the Examiner to provide support for the assertion of what "An artisan of ordinary skill in the art would have been motivated to seek . .," (per MPEP 2144.03.C), the Examiner has not provided any such support. Thus, for at least this reason alone, the alleged motivation cannot be used by the Examiner as alleged support for an obviousness rejection.

Furthermore, Bershteyn recognizes that there is an issue with simulation speed (at least, relative to its particular disclosed circumstances) and, by its very terms, suggests what it considers to be a complete solution. This would discourage skilled artisans from the taking the path which the Examiner contends an artisan "would have been motivated to seek." See, for example, the "SUMMARY OF THE INVENTION" section at col. 3, lines 29-54. There, it is clearly stated that "The Subject Debugger achieves a 50-100 speed advantage over prior debuggers by being able to ignore most of the instructions which would normally be simulated." (col. 3, lines 29-31). It is further stated that "In summary, the speed of the subject hardware-software debugger is markedly increased through the use of high speed simulators which ignore all systems operations except those where design errors are expected to manifest themselves . . . ." (col. 3, lines 45-48). . Nothing in Bershteyn (or any other reference or knowledge properly supported by the Examiner) suggests the "solution" of parallel execution proposed by the Examiner.

Furthermore, the Examiner's basic premise is improper that "multitasking" or "multithreading" is equated with parallel execution. "Multitasking" of processes does not necessarily speed up execution. Rather, "multitasking" or "multithreading," in general, accomplishes what may be considered in some circumstances to be a "fair" (or perhaps need-based) allocation of resources to the various processes. Overall speed of execution need not be directly increased. Thus, not only is the Examiner's assertion of motivation wholly unsupported and contrary to the teachings of the cited references as a procedural matter, the Examiner's assertion of motivation is also substantively flawed.

F) Conclusion

In view of the forgoing, it is respectfully submitted that the subject matter of the appealed claims are present in the combination of cited references, that the Examiner's assertions do not meet the requirements of a prima facie case of obviousness, and that the Examiner cites and improper motivation to make the asserted combination, Therefore, the Examiner's rejection of the appealed claims is erroneous. Accordingly, it is respectfully requested that the pending rejections of all of the claims be reversed.

Respectfully Submitted,
BEYER WEAVER & THOMAS, LLP

[illegible]

Alan S. Hodes
Reg. No. 38,185

P.O. Box 70250
Oakland, CA 94612-0250
(650) 961-8300

VIII. CLAIMS APPENDIX

CLAIMS ON APPEAL

Claim 1 (previously presented): A computer program product for use with a computer system to execute a simulation, comprising:

a plurality of service computer readable program code means, the service program code means configured to collectively determine simulated attributes of objects of an environment under simulated operation wherein the objects are mechanical systems operating in the environment, each service program code means associated with at least a subset of object attributes in an object database and each service program code means executing at a rate independent of the other service program code means wherein the rate is based on the simulated attributes, at least some of the service program code means including computer readable program code means to access and operate upon object attributes, from the object database, with which the service program code means is associated;

write queue computer readable program code means associated with each service program code means that queues write requests from the service program code means to write determined simulated attributes to the object database; and

node computer readable program code means that coordinates execution of the queued requests to cause the determined simulated attributes to be written to the object database in a manner such that each service program code means has a coherent view of all the object attributes.

Claim 2 (original): The program product of claim 1, wherein

the node program code means includes computer readable program code means for creating an image of at least a portion of an object whose attribute is to be written to the object database and for writing the determined simulated attributes to the image; and

to write the determined simulated attributes of the object to the object database, the node program code means associates the image with the object database.

Claim 3 (original): The program product of claim 2, wherein:

the node program code means associates the image with the object database by changing a pointer for the object in the object database to point to the image.

Claim 4 (original): The program product of claim 2, wherein:

the node code program means includes computer readable program code means for notifying at least some of the service program means that the node program code means is associating an image with the object database.

Claim 5 (original): The program product of claim 4, wherein:

in response to a service program code means receiving an object database association notification from a node program code means, the write queue program code means associated with that service program code means queues a request to the node program means to synchronize that service program code means to the image, and

the node program code means includes computer readable program code means for synchronizing the service to the image.

Claim 6 (original): The program product of claim 1, wherein:

the service program code means each include a computational phase during which it operates upon the object attributes and during which the write queue program code means queues the write requests generated by the service program code means during the computational phase; and

the write requests queued for a particular service program code means are processed by the node program code means during a write request processing phase that is outside the computational phase.

Claim 7 (original): The program product of claim 4, wherein the at least some of the service program code means which the node program code means notifies includes service program code means that are associated with the object attributes represented by the node image.

Claim 8 (previously presented): A method executed by a computer to accomplish a simulation, comprising:

a plurality of service steps that collectively determine simulated attributes of objects of an environment under simulated operation wherein the objects are mechanical systems operating in the environment, each service step associated with at least a subset of object attributes in an object database and each service step executing at a rate independent of the other service steps wherein the rate is based on the simulated attributes, at least some of the service steps including steps to access and operate upon object attributes, from the object database, with which the service step is associated;

a write queue computer step associated with each service step that queues write requests from the service step to write determined simulated attributes to the object database; and

a node step that coordinates execution of the queued requests to cause the determined simulated attributes to be written to the object database in a manner such that each service step has a coherent view of all the object attributes.

Claim 9 (original): The method of claim 8, wherein

the node step includes a step of creating an image of at least a portion of an object whose attribute is to be written to the object database and of writing the determined simulated attributes to the image; and

to write the determined simulated attributes of the object to the object database, the node step associates the image with the object database.

Claim 10 (original): The method of claim 9, wherein:

the node step associates the image with the object database by changing a pointer for the object in the object database to point to the image.

Claim 11 (original): The method of claim 9, wherein:

the node step includes a step of notifying at least some of the service steps that the node step is associating an image with the object database.

Claim 12 (original): The method of claim 11, wherein:

in response to a service step receiving an object database association notification from a node step, the write queue step associated with that service step queues a request to the node step to synchronize that service step to the image, and

the node step includes a step of synchronizing the service to the image.

Claim 13 (original): The method of claim 8, wherein:

the service steps each include a computational phase during which it operates upon the object attributes and during which the write queue step queues the write requests generated by the service step during the computational phase; and

the write requests queued for a particular service step are processed by the node step during a write request processing step that is outside the computational step.

Claim 14 (previously presented): The method of claim 11, wherein at least one or more of the service steps which have been modified by the node step as recited in claim 11, are associated with the object attributes represented by the node image.